Unit 4:- File Handling

A program to accept Serial No., Name and Salary of an employee and write it to a file and read from the file.

```
#include <stdio.h>
#include <conio.h>
    struct emp {
                int slno;
                char name[20];
                float sal;
              };
void main()
 {
        struct emp ex;
        float f;
        FILE *fl; clrscr();
  printf("Name Please :");
    gets(ex.name);
  printf("\n Serial Number :");
    scanf("%d", &ex.slno);
    printf("\n salary please :");
     scanf("%f", &f);
       ex.sal = f;
 printf("Serial No. = %d \n Employee Name = %s
          \n Salary = %f \n", ex.slno, ex.name,
                              ex.sal);
    fl= fopen("data.txt", "a");
    if(fl != NULL)
      {   fwrite(&ex, sizeof(ex), 1, fl);
```

```
        printf ("\n File Written Successfully...\n");
        fclose(fl);
    } else
            printf("File Not Opened\n");
    fl= fopen("Data.txt", "r");
      if (fl != NULL)
      {
          printf(" Serial No.\t Name \t\tSalary\n");
    while ( (fread(&ex , sizeof(ex), 1 , fl))>0)
      {
    printf("%d \t %s \t\t %f \n", ex.slno,
                        ex.name, ex.sal);
      }fclose(fl);
    }
    else {
          printf("File Not Opened \n");
          }
  getch();
}
```

To work with secondary storage, FILE is a pre-defined structure in C.
It have several functions to work with files, which include:

naming a file
opening a file
reading from a file
writing to a file and
closing a file

There are two ways to perform file operations in C. One method is known as low-level I/O which uses Unix System calls and the other method is known as high-level I/O operation which uses functions in C's standard I/O library.

Library Functions for File I/O :—

fopen() — used to open a file for formatted I/O and returns a file pointer on success or NULL on failure.

Prototype :—

```
FILE *fopen(const char *filename,
            const char *mode);
```

Where mode for opening a file is

r — open for reading

w — open for writing, discard any previous contents.

a — append mode, write after any previous contents.

r+ — Open file for update (read and write).

w+ — Open file for update (discard any previous data).

a+ — open file for update (append after previous data).

fclose() — closes a file which has been opened for use.

Syntax:- fclose(pointer to file);

getc() — Reads a character from a file.

It is simplest I/O function for FILE structure. These are analogous to getchar() and putchar() functions and handle one character at a time.

putc(c, fp1); writes the character contained in character variable c to the file associated with FILE pointer fp1.

c = getc(fp2); would read a character from the file whose file pointer is fp2.

The file pointer moves by one character position for every operation of getc and putc.

The getc will return an end-of-file marker EOF, when end of file has been reached.

fprintf() — writes a set of data values to a file.

fscanf() — Reads a set of data values from a file.

Syntax:-

fprintf(fp, "control string", list);

where fp — file pointer

control string → contains output specifications for the items in the list.

list → variables, constants and strings

ex:- fprintf(f1, "%s %d %f", name, age, 7.5);

Syntax:- fscanf(fp, "control string", list);

example:- fscanf(fp, "%s %d", item, &qty);

fscanf also returns the number of items that are successfully read. When the end of file is reached, it returns the value EOF.

Example:- Write a prog. to read data from keyboard write it to a file and again read the contents from file and display it on screen.

```
#include <stdio.h>
#include <conio.h>
void main()
{
        FILE *f1;
        char c;
    printf("Enter data to file and press(ctrl+z)");
  f1 = fopen("input", "w");
      while( (c=getchar()) != EOF)
        { putc(c, f1); }
    fclose(f1);
    printf("\n Data Read from file: \n");
      f1 = fopen("input", "r");
       while( (c=getc(f1)) != EOF)
        { printf("%c", c); }
      fclose(f1);
  getch();
}
```

getw() and putw() functions →

The getw and putw are integer-oriented functions, similar to gete and pute functions and are used to read and write integer values.

Syntax is :-

```
putw (integer , fp);
getw (fp);
```

Q Write a program to make a data file which has series of integer numbers. Also read these numbers and then write odd numbers to a file named 'odd' and all even numbers to a file called 'even'.

```
#include <stdio.h>
#include <conio.h>
    void main()
     {    FILE *f1, *f2, *f3;
          int number, i; clrscr();
printf ("Contents of Data file :\n\n");
  f1 = fopen ("data", "w");
   for (i=1; i<=30; i++)
     {   scanf ("%d", &number);
         if (number == -1) break;
         putw (number, f1);
      }
    fclose (f1);
f1 = fopen ("data", "r"); f2 = fopen ("odd", "w");
f3 = fopen ("even","w");
while ((number = getw(f1)) != EOF)
  {     if (number % 2 == 0) putw (number, f3);
        else
          putw (number, f2);
  }
```

```
    fclose(f1); fclose(f2); fclose(f3);
    f2 = fopen("odd", "r");
    f3 = fopen("even", "r");
    printf("\n\n Contents of odd file \n\n");
  while ((number = getw(f2)) != EOF)
    { printf("%4d", number); }
  printf("\n\n Contents of even file \n\n");
  while ((number = getw(f3)) != EOF)
    { printf("%4d", number); }
    fclose(f2); fclose(f3);
  getch();
}
```

Q. Write a program to open a file inventory and store in it the following data :-

Item name	Number	Price	Quantity
AAA-1	111	17.50	115
BBB-2	125	36.00	75
C-3	247	31.75	104

Extend the program to read this data from the file inventory and display the inventory table with the value of each item.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    FILE *fp;
    int number, quantity, i;
    float price, value;
    char item[10], filename[10];
    clrscr();
```

```
printf ("Enter a filename : ");
scanf ("%s", filename);
fp = fopen (filename, "w");
printf ("Input Inventory Data Now :\n\n");
printf ("Item Name  Number  Price  Quantity \n");
  for (i=1; i<=3; i++)
   {   fscanf (stdin, "%s %d %f %d",
                item, &number, &price, &quantity);
fprintf (fp, "%s %d %.2f %d", item,
                number, price, quantity);
   }
fclose (fp);
fprintf (stdout, "\n\n");
fp = fopen (filename, "r");
printf ("Item Name  Number  Price  Quantity  Value\n");
for (i=1; i<=3; i++)
 {
   fscanf (fp, "%s %d %f %d", item, &number,
                &price, &quantity);
   value = price * quantity;
   fprintf (stdout, "%-8s %7d %8.2f %8d
            %11.2f \n", item, number, price,
            quantity, value);
 }
fclose (fp);
getch ();
}
```

---

ftell function takes a file pointer and returns a number of type long, that corresponds to the current position

The function is useful in saving the current-position of a file.

Q Write a program to append additional items to the file inventory created before and print the total contents of the file.

```
#include <stdio.h>
#include <conio.h>
  struct  invent_record
    {
        char name[10];
        int number;
        float price;
        int quantity;
    };
void  main()
  {     typedef struct invent_record ir;
        ir item;
        char filename[10];
        int response;
        FILE *fp;
        long n;
    void append(ir *x, FILE *y);
    clrscr();
    printf("\n Type the filename:");
     scanf("%s", filename);
     fp = fopen(filename, "a+");
       do {
                append(&item, fp);
       printf("\n Item %s appended.\n", item.name);
       printf("\n Do you want to add more(1-yes,
                                    0-no)?");
       scanf("%d", &response);
          } while(response==1);
     n = ftell(fp);
     fclose(fp);
```

```
fp= fopen (filename, "r");
while (ftell (fp) < n)
{
    fscanf (fp, "%s %d %f %d", item.name,
    &item.number, &item.price, &item.quantity);
fprintf (stdout, "%-8s %7d %8.2f %8d\n",
        item.name, item.number, item.price,
        item.quantity);
}
fclose (fp);
getch ();
}       //close of main

void append (ir *product, FILE *ptr)
{
    printf ("Enter Item Name : ");
        scanf ("%s", product -> name);
    printf ("\n Enter Item Number : ");
        scanf ("%d", &product -> number);
    printf ("\n Enter Item Price : ");
        scanf ("%f", &product -> price);
    printf ("Quantity : ");
        scanf ("%d", &product -> quantity);
    fprintf (ptr, "%s %d %.2f %d", product->name,
    product -> number, product -> price,
    product -> quantity);
}
```

int fgetc (FILE *fp);
int fputc (int c, FILE *fp);

fgetc gets a character from a stream

fputc outputs a character to a stream

fgetc and fputc are the function version of getc and putc macro.

int putch (int ch);

putch outputs the character ch to the current text window.

int getchar(void);
int putchar (int c);

getchar reads from std. input device.
It has '\n' for Return
EOF for End of file Symbol (Ctrl+Z).

int fseek (FILE *fp, long offset, int whence);

Repositions the file pointer of a stream.

*fp is a pointer to FILE type
offset — current pointing position from 0 to onward.

Whence — 0, 1, 2

Seeks from beginning, current, end position.

rewind (FILE *fp) function takes a file pointer and resets the position to the start of the file.

ftell takes a file pointer and return a number of type long, that corresponds to the current position. This function is useful in saving the current position of a file, which can be used later in the program.

```
n = ftell (fp);
```

here n has current position of file pointer. When file is opened, a rewind is done implicitly in case of reading and writing mode.

Write a program that uses the functions ftell and fseek.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    FILE *fp;
    long n; char c; clrscr();
  fp = fopen ("Random", "w");
      while ((c = getchar()) != EOF)
        putc(c, fp);
    printf ("No. of characters entered = %ld\n",
                                ftell (fp));
    fclose (fp);
  fp = fopen ("Random", "r");
     n = 0L;
   while (feof (fp) == 0)
    {
      fseek (fp, n, 0); //goto the beginning
      printf ("Position of %c is %ld \n",
                        getc(fp), ftell (fp));
      n = n + 5L;
    }
```

```
        putchar('\n');
        fseek(fp, -1L, 2);
        do{
            putchar(getc(fp));
        }while(!fseek(fp, -2L, 1));
        fclose(fp);
    getch();
}
```

Or

```
//searching record from a file
#include<stdio.h> #include<conio.h>
typedef struct{ int n; char nm[10]; float s;}emp;
void main()
{ emp e1; int n, p; FILE *f; clrscr();
 f=fopen("data.txt","r");
 if(f != NULL)
  {
     printf("\n Enter Record Number:\n");
     scanf("%d",&n);   p=sizeof(e1)*(n-1);
     fseek(f, p, 0);
     fread(&e1, sizeof(e1), 1, f);
    printf("%d \t %s \t %f \n", e1.n, e1.nm,
                                e1.s);
    fclose(f);
  } else { printf("File Not Opened \n"); }
  getch();
}
```

Unit-5 Bitwise operators:

C language has the unique feature as compared to other high-level languages of allows direct manipulation of individual bits within a word. Bit-level manipulations are used in setting a bit to 1 or 0.

C supports the following operators:

1. Bitwise logical operators
2. Bitwise shift operators
3. One's complement operator

All these operators work only on integer type operands.

(1) Bitwise Logical Operators: —

& — AND

| — OR

^ — Exclusive OR

operand1	operand2	op1 & op2	op1 | op2	op1 ^ op2
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

(2) Bitwise Shift operators: — Used to move bit patterns either to the left or to the right.

Left Shift << (op << n)

Right Shift >> (op >> n)

x — 01001001 11001011

x<<3 — 01001110010 1000

x>>3 — 000 01001001 11001

Shift operators are often used for multiplication and division by powers of two.

$x = y << 1$

Here, decimal value of x = decimal value of y multiplied by 2

Similarly, $x = y >> 1$

In this case, the value of x will be the value of y divided by 2.

The shift operators, when combined with the logical bitwise operators, are useful for extracting data from an integer field that holds multiple pieces of information. This process is known as masking.

Use of masking in converting a number into binary format :—

```
#include<stdio.h> #include<conio.h>
void main()
{
    int a, b, m; count, nbits;
    unsigned mask; clrscr();
  nbits = 8 * sizeof(int);
  m = 1 << (nbits - 1);
      do {
          printf("Enter a no 0 to stop");
          scanf("%d", &a);
          mask = m;
          for (count=1; count <= nbits;
                              count++)
          {
              b = (a & mask) ? 1 : 0;
              printf("%d", b);
              if(count % 4 == 0)
              printf(" ");
              mask >>= 1;
}
```

```
    } while (a != 0);
}
```

Function → A function is a self contained block of code that performs a particular task. Once a function has been designed and packed, it can be treated as a 'black-box' that takes some data from the main program and returns a value.

⇒ Any function can call any other function. In fact, it can call itself

⇒ A called function can also call another function.

⇒ A function can be called more than once.

⇒ This is one of the main features of using functions.

⇒ The functions can be placed in any order.

⇒ A called function can be placed either before or after the calling function.

⇒ However, it is the usual practice to put all the called functions at the end.

The form of C functions →

The default return type of function in C is int.

```
return-type functionname (argument list)
{
  works;
  return (expression);
}
```

Here, argument list is optional

⇒ A function may have more than one return statements based on a certain conditions.

⇒ When the compiler of C encounters a function call, the control is transferred to the function definition.

⇒ However, a function cannot be used on the left side of an assignment statement.
Ex:- mul(a, b) = 15; is invalid statement.

category of functions →

(a) Function with no arguments and no return value
(b) Function with arguments and no return value
(c) Function with arguments and return value.

Example of using command line argument with FILE :—

Q Write a program that will receive a filename and a line of text from command line argument and write the text to the file.

Ans
```
#include<stdio.h>
#include<conio.h>

void main(int argc, char *argv[])
{
    FILE *fp; int i; char word[30];
    clrscr();
  fp = fopen(argv[1], "w");
printf("\n No. of arguments in command
                line = %d \n\n", argc);

for (i=2 ; i < argc ; i++)
      fprintf (fp, "%s", argv[i]);
  fclose (fp);
```

```
printf("Contents of %s file \n\n", argv[1]);
fp = fopen(argv[1], "r");
for (i=2 ; i<argc ; i++)
{
    fscanf(fp, "%s", word);
    printf("%s", word);
}
fclose(fp);
printf("\n\n");
for(i=0 ; i<argc ; i++)
    printf("%.*s\n", i*5, argv[i]);
getch();
}
```

(8) Write a program in C to make an array of numbers, sort it and search an element after sorting the array.

```
#include<stdio.h>
#include<conio.h>
void sort(int *);
int search(int *, int);

void main()
{
    int arr[10], s, i; clrscr();
    printf("\n Enter any 10 Numbers :");
    for (i=0; i<10; i++)
    {
        scanf("%d", &arr[i]);
    }
printf("\n The Elements of array are :\n");
for(i=0; i<10; i++)
    printf("%4d", arr[i]);
sort(arr);
```

```
    printf("\n Elements After Sort : \n");
    for (i=0; i<10; i++)
      printf ("%4d", arr[i]);
    printf("\n Enter ano- to search occurrence:");
    scanf("%d", &s);
    printf("\n The Number %d is found : %d
                    times", s, search(arr, s));
    getch();
}

void sort (int *a)
{
    int i, j;
    for (i=0; i<10; i++)
    {
        for (j=0; j<9; j++)
        {
            if (a[j] > a[j+1])
            {
                int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
}

int search (int *a, int e)
{
    int pos=0, i;
```

```
for (i=0; i<10; i++)
{
    if(arr[i] == e)
        pos++;
}
return pos;
}
```

Sorting Concept → The most common process of arranging data according to their values is known as sorting.

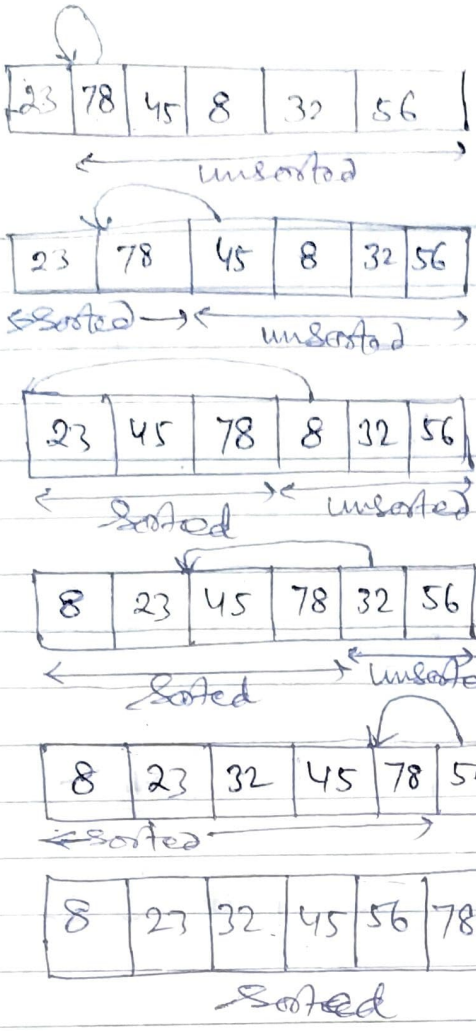
Sort is broadly categorized in two types — Internal and External (merges)
Internal is further of three types — Insertion, Selection and Bubble or Exchange

→ Internal sort is a sort in which all of the data are held in primary memory during the sorting process.
→ External sort uses primary memory for the data currently being sorted and secondary storage for any data that will not fit in primary memory.

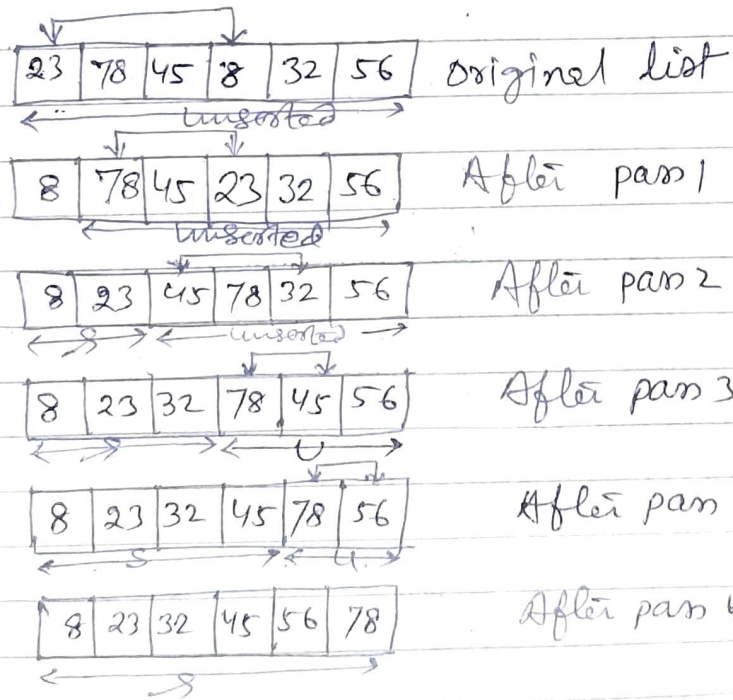
Internal sort is of three types :—

Insertion Sort →

~~algorithm insertSort(ref list <array> val <index>)~~

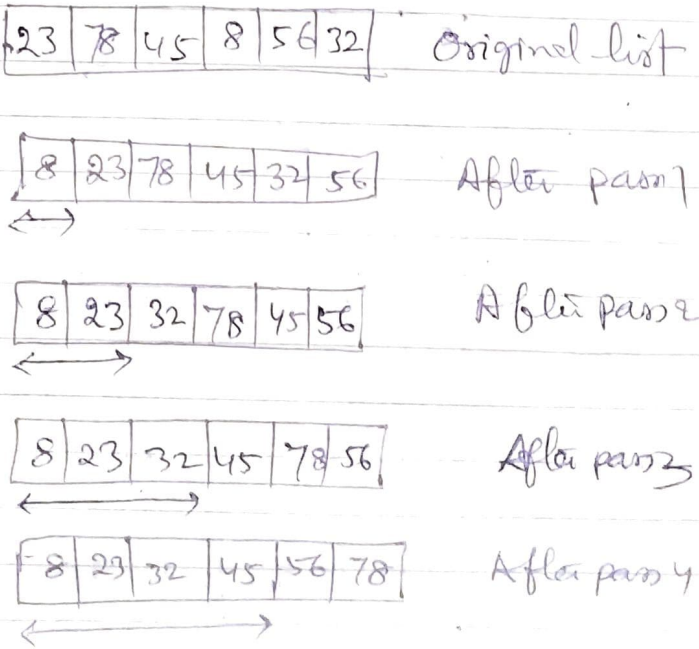


Selection Sort →



Heap Sort →

Bubble Sort →



Quick Sort → It is an exchange sort in which a pivot key is placed in its correct position in the array while rearranging other elements widely dispersed across the list.

① Write a C program to make a function using pointer type argument — to accept any two numbers from user, write the table of that numbers and also calculate the difference between corresponding numbers in table.

```
#include<stdio.h>
#include<conio.h>
void table(int *, int *);
void main()
{
    int a, b;
    clrscr();
    printf("Enter any two Numbers :\n");
    scanf("%d %d", &a, &b);
    table(&a, &b);
    getch();
}
void table(int *a, int *b)
{
    int i, m, n, c;
    printf("\nTable of %d and %d and
        corresponding difference are :\n", *a, *b);
    printf("First Number---Second No.---Difference\n");
    for(i=1; i<=10; i++)
    {
        m = *a * i;
        n = *b * i;
        c = (m>n)? (m-n) : (n-m);
        printf("%d\t %d\t =%d\n", m, n, c);
    }
}
```

W.A.P. to store inventory data in file & get from file.

```
void main() { FILE *fp; long n; int no, qty, i,
              choice=1; float price, value;
char item[15], filename[15]; clrscr();
printf("\n Enter a file name :"); scanf("%s", filename);
fp = fopen(filename, "w");
do { printf("\n Input a Inventory Data Now :\n");
     Printf("\n Item_Name Number Price Quantity \n");
  fscanf(stdin, "%s %d %f %d", item, &no, &price
                               &qty);
  fprintf(fp, "%s %d %.2f %d", item, no, price,
                               qty);
printf("\n Do you want to add more (1-yes, 0-no)");
  scanf("%d", &choice);
  }while(choice==1);
  n = ftell(fp);
  fclose(fp);
  fprintf(stdout, "\n\n");
  fp = fopen(filename, "r");
  printf("Item Name Number Price Quantity
                               Value \n");
  while(ftell(fp) < n)
  {
    fscanf(fp, "%s %d %f %d", item, &no,
                &price, &qty);
      value = price * qty;
    fprintf(stdout, "%-3s %7d %8.2f %3d
                %11.2f \n", item, no,
              price, qty, value);
  } fclose(fp); getch(); }
```

Q. Write a program to add some data to a file and also after end of file symbol, store it in file and also print the content of file on screen.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    FILE *fp;
    int c; long count=0;
    printf("\n Enter Data to store it in file:\n");
    fp= fopen("str.txt","w");
    while((c=getchar()) != EOF)
    { fputc(c, fp);
    }
    fclose(fp);
printf("\n\n The Contents read from file:\n\n");
    fp= fopen("str.txt","r");
    while( (c=fgetc(fp)) != EOF)
    {
        printf("%c", (char)c);
        ++count;
    }
    fclose(fp);
printf("\n The no. of characters from file is: %ld",
                                    count);
    getch();
}
```

Some functions of filing →

fflush() — is used to flush any buffered data out of an output stream. Output stream may be buffered or unbuffered. With buffered output, as data is written to the stream, the O/S saves this data to an intermediate buffer.

fflush() forces the buffer to be written out to the associated file.

Syntax: int fflush(FILE *stream);

It returns, 0 on succes & EOF on failure.

fread() → used to read data from file. It returns the no. of characters read if you passed it a character array.

Syntax:- int fread(char *s, int size, int n, FILE *fp);

Where size — sizeof character pointer s in bytes,

n — no. of data read from fp.

fp — file pointer

s — data stream read from fp.

eg:- while((fread(&ex, sizeof(ex), 1, fp) > 0)
{ printf("%s %d %f", ex.nm, ex.age, ex.sal);
}

fwrite() → Used to write data to file. It also returns integer type data.

Syntax:- int fwrite(char *s, int size, int n, FILE *fp);

where size — size of *s in bytes
n — no. of data items to be written
fp — file pointer
s — data stream to be written on fp.

eg:- fwrite(&ex, sizeof(ex), 1, fp);

generally, reference of structure object is used in real world programming for the first argument of fread and fwrite function.

Searching Record from a file :—

```
#include<stdio.h>
#include<conio.h>
typedef struct{char name[15];
               int number;
               float price;
               int quantity;
              }inventory;

void main()
{   FILE *fp;
    inventory item; int l;
    char filename[10]; char ch;
float value; long n; float p; clrscr();
    printf("\n Type the filename:");
    scanf("%s", filename);
  fp = fopen(filename, "a+");
  do{ printf("\n Enter Item Name:");
      scanf("%s", item.name);
      printf("\n Enter Item Number:");
      scanf("%d", &item.number);
```

```
printf("\n Enter price of Item: ");
scanf("%f", &p);
item.price = p;
printf("\n Enter Quantity :");
scanf("%d", &item.quantity);
value = item.price * item.quantity ;
fprintf(fp, "%s %d %.2f %d %.2f",
        item.name, item.number, item.price,
        item.quantity, value);
printf("\n Item %s appended \n", item.name);
printf("\n Do you want to add more (Y/n): ");
scanf("%c", ch);
} while (ch == 'y');
        n = ftell(fp);
        fclose(fp);
fp = fopen(filename, "r");
    while ( ftell (fp) < n)
      {
fscanf(fp, "%s %d %f %d %f", item.name,
 &item.number, &item.price, &item.quantity, &value);
fprintf(stdout, "%-8s %7d %8.2f %8d %8.2f",
    item.name, item.number, item.price, item.quantity,
    value);
      }
fclose(fp);
fp = fopen(filename, "r");
    if (fp != NULL)
    {   printf("\n Enter Record No to find :");
```

```
    scanf("%d", &n);
    l = sizeof(item) * (n-1);
      fseek(fp, l, 0);
      fread(&item, sizeof(item), 1, fp);
printf("%s \t %d \t%.f \t%d \t%.f \n",
  item.name, item.number, item.price,
  item.quantity, value);
}
    fclose(fp);
    else
    { printf("\n File Not Opened :\n");
    }
  fclose(fp);
  getch();
}
```

Matrix Multiplication Program

```
#include<stdio.h>
#include<conio.h>
void main()
{  int i, j, k;
   int a, b, c; //variables to get row & column of two matrix
   int sum;
   int m[10][10], n[10][10], l[10][10]; clrscr();
   printf("\n The first Matrix:\n");
   printf("\n Enter No. of Rows and Columns for first matrix:
                                                \t");
   scanf("%d %d", &a, &b);
   printf("The second Matrix:\n");
   printf("\n Enter No. of Rows and Columns for second matrix:
                                                \t");
   scanf("%d %d", &b, &c);
/* Note: For matrix multiplication the no. of Columns in the first
   matrix should be equal to the no. of rows in the second
   matrix */
```

```
printf("\n Enter the values of the first matrix :\n");
for (i=0; i<a; i++)
{
    for(j=0; j<b; j++)
    {
        scanf("%d", &m[i][j]);
    }
}
printf("\n Enter the values of the second matrix :\n");
for( i=0; i<b; i++)
{  for (j=0; j<c; j++)
    {  scanf("%d", &n[i][j]);
    }
}
        for(i=0; i<a; i++)
        {  for(j=0; j<c; j++)
            {  sum=0;
               for(k=0; k<b; k++)
               {
                  sum+=(m[i][k] * n[k][j]);
                  l[i][j] = sum;
               }
            }
        }
printf("The multiplied matrix is :\n");
for(i=0; i<a; i++)
{
    for(j=0; j<c; j++)
    {
        printf("%d \t", l[i][j]);
    }
    printf("\n");
}
getch();
} //close of main function
```